Lab-13

This assignment demonstrates the training and evaluation of a neural network for fertility classification using a single hidden layer. It covers key concepts such as feedforward pass, backpropagation, and error calculation. You will be implementing single hidden layer neural networks from scratch.

**Instructions:**

* Submit the graph along with the complete code (and code+output pdf). Make sure all the print statements mentioned in the document are clearly printed and visible.
* Using Google Colab is **recommended** (not necessary) for this lab due to tensorflow installation issues.

**Step1:** Import libraries

* Numpy
* Pyplot
* Mean Squared Error
* Tensorflow

**Step2**: Load “Fertility\_Diagnosis.txt” file.

* Use np.genfromtxt()
* Create training and testing datasets.
* The last column in this dataset is the output column. The rest of the columns are input features.

**Step3**: Define input layer size and output layer size.

* Create an empty list to store training error and testing error separately.
* Keep error tolerance = 0.05.

**Step4**: We will be creating 9 different neural networks and monitoring the train and test error for each network. Each network only differs in hidden layer size which ranges from 1 to 9. For each network, do the following-

* Initialize random synapse weights.
* Define feedforward network for training data.
  + layer 0 – input data
  + Layer 1- Sigmoid activation Function
  + Layer 2- Sigmoid activation Function
* Check for accuracy.
  + Calculate mean square error.
  + Print MSE
* Keep learning rate (alpha) = 0.001
* Print – training with alpha value.

**Step5**: For number of epochs ranging from 1 to 1000001 perform backpropagation.

* Define feedforward network as in step 4.
* Define cost function (Difference between training data outputs and layer 2)
* Calculate the derivative of the sigmoid activation with respect to the weighted sum of inputs at layer 2. This derivative can be used in backpropagation algorithm for updating the weights.
* Update the weight using calculated derivative.
* Now calculate the derivative of the sigmoid activation with respects to the weighted sum of inputs at layer 1.

**Step6**: Check the convergence and print the debug data.

* If error value is less than error tolerance value, break the loop and print the number of epochs and error at which training is stopped.
* Print the debug data during training for every 100000 epochs.
* If no convergence happens within max epochs(1000001), print “model did not converge”.

**Step7**: Evaluate the training results.

* Perform a feedforward pass through the trained network on the training data.
* Calculate and print the mean squared error for the training dataset.
* Append the training error to the list.

**Step8**: Evaluate testing results.

* Perform a feedforward pass through the trained network on the testing data.
* Calculate and print the mean squared error for the testing dataset.
* Append the testing error to the list.

**Step9**: Plot the results.

* Plot the training and testing errors for different numbers of neurons in the hidden layer using Matplotlib.
* Reference output given below

A graph of a graph with lines and numbers

Description automatically generated with medium confidence